SYSTEMS AND METHODS THAT AUTHORIZE TARGET DEVICES UTILIZING PROPRIETARY SOFTWARE AND/OR HARDWARE

Field of the Invention

The present invention relates to integrated systems and methods of operating same, and more particularly, to integrated systems requiring authorization to utilize proprietary software and/or hardware.

Background of the Invention

Programmable logic devices (PLDs), such as field programmable gate arrays (FPGAs), typically utilize binary configuration data when performing desired operations in a target application. Frequently, an end user of a programmable logic device would acquire proprietary configuration data ("software") from a software developer by executing a software license agreement. Pursuant to this agreement, an end user would typically receive either design source code or more often the binary configuration data compiled from the source code. As illustrated by the conventional operations **10** of FIG. 1, proprietary design source code of a developer, Block **12**, is typically provided to an end user by compiling or processing the source code into binary configuration data, Blocks **14**, **16**. An end user of a PLD would typically then download the configuration data into a programmable read only memory (PROM) mounted on a printed circuit board, Block **18**. The configuration data in this memory could then be accessed and loaded into a PLD during start-up operations, Block **20**, so that the PLD becomes configured by the configuration data.

Unfortunately, these operations may put the software developer at risk that unauthorized copies of the binary configuration data might be used in additional target applications for which the software developer does not receive compensation. To address this possibility, software developers may develop complex software license agreements to limit unauthorized copying. Such agreements may be difficult to negotiate and may require large up-front royalty fees. Moreover, such terms may preclude users from entering such licenses when only relatively few applications for the proprietary software are anticipated. Thus, notwithstanding conventional licensing techniques for incorporating proprietary software into programmable logic devices, there continues to be a need for improved techniques that do not suffer from the aforementioned limitations associated with conventional software licensing.

Summary of the Invention

Preferred integrated systems include devices that authorize programmable logic devices to operate under at least partial control of proprietary software. Each of these "authorization" devices preferably provides continuous or at least periodic authorization to a respective programmable logic device while it is operating in a desired application. This continuous or periodic authorization is preferably provided only so long as the version of software being used by the programmable logic device matches the version of software the authorization device was designed to evaluate and approve.

According to a first preferred embodiment of the present invention, an integrated system comprises an authorization device that generates an encrypted data stream and a programmable logic device (PLD) that also generates an encrypted data stream while simultaneously operating under at least partial control of program code during a first time interval. This program code or software may take the form of data that configures circuitry within the programmable logic device.

Authorization detection circuitry is also preferably provided within the programmable logic device. This circuitry compares the encrypted data streams at least periodically during the first time interval. This circuitry may also disable operation of the programmable logic device if the encrypted data streams indicate that the programmable logic device is not authorized to use the program code. Disabling operation of the programmable logic device may constitute a complete shut down of the programmable logic device or the performance of the programmable logic device may be degraded or impaired sufficiently to render it unacceptable in the desired application.

In particular, the encrypted data streams are evaluated at least periodically during the first time interval to determine whether a "match" is present between the authorization device and the proprietary software used to configure the programmable logic device. A direct ongoing comparison can be made between the encrypted data streams to determine whether there is a sufficiently close identity therebetween while the programmable logic device is operating in a target application. If a sufficiently close identity is present, a "good" flag may be generated within the programmable logic device to enable proper operation for at least some limited time period. An exact identity between the encrypted data streams is preferably not required by the authorization detection circuitry in order to maintain the status of the good flag. However, if a sufficiently close identity is not present between the encrypted data streams over a threshold period of time, then a "fail" flag may be generated. The generation of this fail flag preferably causes the programmable logic device to enter a disabled state. In this disabled state, the programmable logic device may cease to operate or may operate at a degraded or impaired performance level caused by the intentional internal generation of operating errors (e.g., "random" operating errors) by circuitry within the programmable logic device. Other degraded performance states that make the device unfit for the target application may also be possible.

The above-described authorization scheme may also be applied to other forms of programmable logic devices. Some of these programmable logics devices are frequently referred to by the acronyms PLDs, PLAs, PALs, FPLAs, EPLDs, EEPLDs, LCAs, and FPGAs. In addition, the preferred authorization scheme may be applied to application-specific integrated circuits (ASICs) that perform operations which are exclusively or at least partially hardware based. For example, an ASIC may be designed to perform a plurality of functions and operations useful for a variety of applications. Customers purchasing such ASICs may be able to upgrade or expand the functions and operations performed by the ASIC by purchasing one or more authorization devices at the time the ASIC is purchased or thereafter.

Additional embodiments of the present invention include preferred methods of operating programmable logic devices. These methods preferably include the steps of generating the encrypted data streams during a first time interval while simultaneously operating the programmable logic device under at least partial control of program code that may constitute configuration data. These encrypted data streams are preferably evaluated periodically during the first time interval. The operation of the programmable logic device is then disabled during a second time interval if a comparison of the data streams indicate that the programmable logic device is unauthorized to use the program code.

Brief Description of the Drawings

FIG. 1 is a flow diagram of conventional operations used to load configuration data into programmable logic devices.

FIG. 2 is a flow diagram of exemplary operations used to load configuration data into programmable logic devices.

FIG. 3A is a block diagram of a first integrated system comprising a programmable logic device (PLD) and an authorization device according to a first embodiment of the present invention.

FIG. 3B is a block diagram of a second integrated system comprising an application-specific integrated circuit (ASIC) and an authorization device according to a second embodiment of the present invention.

FIG. 3C is a block diagram of a third integrated system comprising PLDs, ASICs and authorization devices according to a third embodiment of the present invention.

FIG. 4A is a block electrical schematic of preferred deadman circuitry within a PLD or ASIC according to the present invention.

FIG. 4B is a block electrical schematic of a preferred authorization device according to the present invention.

FIG. 4C is a block diagram illustrating operations performed by the stream encryptor of FIG. 4B.

FIG. 5 is a flow diagram of operations that illustrate preferred methods of operating programmable logic devices according to the present invention.

Description of Preferred Embodiments

The present invention will now be described more fully hereinafter with reference to the accompanying drawings, in which preferred embodiments of the invention are shown. This invention may, however, be embodied in different forms and should not be construed as limited to the embodiments set forth herein which are provided as preferred examples. Rather, these embodiments are provided so that this disclosure will be thorough and complete, and will fully convey the scope of the invention to those skilled in the art. Like numbers refer to like elements throughout.

Referring now to FIG. 2, exemplary operations **30** for loading a programmable logic device with program code will be described. In particular, FIG. 2 illustrates an operation to generate proprietary design and "deadman" source code, Block **32**. As described more fully hereinbelow, this "deadman" source code may augment the design source

code so that a programmable logic device performing target operations in accordance with the design code can also be monitored and approved by an authorization device. As illustrated by Blocks **34** and **36**, conventional operations may then be performed to process the source code (including "deadman" source code) into binary configuration data. The binary configuration data may then be provided by the source code developer to a PLD manufacturer or end user. According to the present invention, this conveyance can be made without the need to secure a software license agreement between the conveying party and the receiving party. Instead, preferred authorization devices may be sold to authorize each copy of proprietary software being used in a target application. As illustrated by Block **38**, the binary configuration data may be loaded into a programmable read-only memory (PROM) or another memory storage device that is mounted on a printed circuit board (PCB) along with a respective PLD. The contents of the PROM may then be accessed and downloaded by the PLD using conventional techniques, Block **40**. The downloaded configuration data may operate to configure devices within the PLD that perform operations designed for the target application and configure devices that operate as deadman circuitry. Alternatively, the PLD may acquire the binary configuration data by accessing a network or other device or system external to the PCB.

Referring now to FIGS. 3A-3C, a plurality of integrated systems according to embodiments of the present invention will be generally described. In particular, FIG. 3A illustrates an integrated system **50** comprising a PROM **52** and a preferred PLD **54** having deadman circuitry **54a** therein that becomes programmed by receiving binary configuration data from the PROM **52**. The system also includes an authorization device **56** that provides at least periodic authorization to the PLD **54** to operate under control of the loaded binary configuration data. In FIG. 3B, another preferred integrated system **50'** is illustrated that comprises an application-specific integrated circuit (ASIC) **58** having

deadman circuitry **58a** therein and an authorization device **56** that is electrically coupled to the ASIC **58**. The deadman circuitry **58a** used in ASIC applications typically performs similar operations to the devices within a PLD that are configured as deadman circuitry by the configuration data. In FIG. 3C, an exemplary integrated system **50"** is illustrated that comprises a PLD **54** and an ASIC **58** on an printed circuit board. A plurality of authorization devices **56a-56c** are also provided. As illustrated, these devices may operate in response to a central controller **66**. Additional devices including a PROM **52**, a PLD **60** not requiring authorization, a memory array **62** and I/O circuitry **64** may also be provided. Thus, an integrated system may have one or more PLDs or ASICs and respective authorization devices that operate in conjunction with conventional hardware.

Referring now to FIGS. 4A-4B, a more detailed description of the operation of the integrated systems of FIGS. 3A-3C will be provided. These systems include devices that authorize operation of programmable and other logic devices that operate under at least partial control of proprietary software and/or hardware. With respect to the programmable logic devices **54** illustrated by FIG. 3A, the authorization device **56** preferably provides continuous authorization to a respective programmable logic device **54** while it is operating in a desired target application. This continuous authorization is preferably provided only so long as the binary configuration data ("software") being used by the programmable logic device **54** matches the version of software the authorization device **56** was designed to evaluate and approve.

According to the preferred deadman circuitry **54a** of FIG. 4A, a first data stream P may be generated by a weak random data generator **70**. As illustrated, the weak random data generator **70** may provide a weak random data stream in response to a clock signal and a noise signal and may be of conventional design. An outgoing data framer **72** and an open-drain driver **78** may also be provided so that the first data stream P can be

passed to an input/output pad (I/O). This first data stream P may be provided in-sync with timing signals generated by a central timing circuit **74**. Other formats for the first data stream P may also be used.

Referring now to the preferred authorization device **56** of FIG. 4B, a second encrypted data stream R is generated and provided by a respective stream encryptor **100** to the input/output pad (I/O). This second encrypted data stream R is preferably provided in response to the first data stream P. The illustrated I/O pads associated with the authorization device **56** of FIG. 4B and the deadman circuitry **54a** of FIG. 4A may be connected together by a single-wire bus. A two-wire bus can also be used and such configuration may eliminate the need to provide open-drain drivers. Alternatively, a three-wire bus can be used with an additional clock and such implementation may eliminate the need for timing/framing pulses. If a single-wire bus is used to provide an electrical connection, then the second encrypted data stream R may be time-division multiplexed with the first data stream P on the single-wire bus using a half-duplex format. Operations for multiplexing data in a single-wire bus are generally known to those skilled in the art and need not be described further herein. The illustrated authorization device **56** may comprise an input buffer **90** and an incoming data sampler **92** that operates in response to a central timing circuit **94**. Using these conventional devices, the first data stream P can be retrieved from the I/O pad and provided to an input of the stream encryptor **100**. As described more fully hereinbelow, the stream encryptor **100** of FIG. 4B generates the second encrypted data stream R (with error) using an encryption operation. This encrypted data stream R is then framed and provided to an I/O pad using conventional devices such as the illustrated outgoing data framer **96** and open drain driver **98**. To increase security and to inhibit the likelihood that the construction of the authorization device **56** can be readily reverse-engineered, circuitry within the illustrated stream encryptor **100** may be designed to randomly insert errors into the second encrypted data stream R. The presence of a limited number of intentional

errors will typically increase the difficulty in determining the encryption operation by evaluating the data on the single-wire bus using conventional reverse engineering techniques. If protection against reverse-engineering is not required, then the data streams need not be encrypted.

Referring again to the deadman circuitry **54a** of FIG. 4A, an input buffer **80** receives the time-division multiplexed stream (e.g., half-duplex stream) and passes the received stream to an incoming data sampler **76**. In response to central timing, this data sampler **76** extracts the second encrypted data stream R from the multiplexed stream and provides it to an authorization detection circuit (ADC) **84**. As illustrated, the ADC **84** may comprise an exclusive OR (XOR) gate, an error history evaluation circuit and an error timer. The deadman circuitry **54a** may also comprise a stream encryptor **82** that generates and provides a third encrypted data stream R' to an input of the XOR gate within the ADC **84**, in response to the first data stream P and the second encrypted data stream R (with error). To reduce the complexity of the ADC **84**, the stream encryptor **82** within the deadman circuitry **54a** preferably performs operations similar to the stream encryptor **100** within the authorization device **56**, but typically need not perform error insertion operations. This third encrypted data stream R' is preferably generated within the deadman circuitry **54a** (under control of the configuration data compiled from the deadman source code) while other portions of the PLD **54** simultaneously operate under at least partial control of the configuration data compiled from the design source code.

According to a preferred aspect of the ADC **84**, a logic 1 error signal is generated at an output of the XOR gate every time a mismatch between the second and third encrypted data streams is detected. A running history of these errors signals is then maintained by the error history circuit. If an insufficient number of errors are detected within a predetermined threshold time period, for example, then a "good" flag may be generated by the ADC **84** to enable proper operation of the PLD **54** for

at least some limited time period. Because of the presence of intentional errors in the second encrypted data stream R, an exact identity between the encrypted data streams is preferably not be required by the ADC **84** in order to maintain the status of the good flag. However, if a sufficiently close identity is not present between the encrypted data streams over a threshold period of time, then a "fail" flag may be generated. The generation of this fail flag preferably causes the PLD **54** to enter a disabled state. In this disabled state, the PLD **54** may cease to operate or may operate at a degraded performance level caused by the intentional internal generation of operating errors (e.g., "random" operating errors) by circuitry within the PLD **54**. Other degraded performance states may also be possible.

Referring now to FIGS. 4A-4C, encryption operations performed by the deadman circuitry **54a** and the authorization device **56** will be more fully described. As described above with respect to the weak random data generator **70** in FIG. 4A, a first data stream P may be generated by mixing noise and clock signals. This mixing operation may be performed using conventional techniques using an "unpredictable" circuit that sequentially generates a weak pseudo-random stream of bit data as $\{P_1, P_2, P_3, ..., P_n\}$, where "n" is an integer. This first data stream P is then provided to a first stream encryptor **82** within the configured deadman circuitry **54a** and also to the authorization device **56**. The authorization device **56** is preferably configured to respond to the first data stream P by generating a second encrypted data stream R of bit data as $\{R_1, R_2, R_3, ..., R_n\}$. This second encrypted data stream R is then fed back, potentially with a limited number of intentional and randomly inserted errors therein, to the deadman circuitry **54a** within the PLD **54**. The second encrypted data stream R may be time division multiplexed (e.g., interleaved) with the first data stream P. Thus, the bit data on the single-wire bus connecting the authorization device **56** to the PLD **54** may look like: $\{P_1, R_1, P_2, R_2, P_3, R_3, ..., P_n, R_n\}$.

The second encrypted data stream R is preferably generated by performing an encryption operation that evaluates the first data stream P and a plurality of previously generated bits in the second encrypted data stream R. As illustrated by the flow diagram of operations shown in FIG. 4C, the second stream encryptor **100** within the authorization device **56** may use conventional permuting operations to sequentially determine a plurality of permuted bits as $\{H_1, H_2, H_3, ..., H_n\}$ during a first time interval, with each permuted bit being determined in accordance with the following expression:

$$H_i = f_p (P_i, R_{i-j}, ... R_{i-j-k})$$

where f_p is a permuting function, "i" and "j" are positive integers and "k" represents a preferred "depth" to which the first data stream R is evaluated. The second stream encryptor **100** within the authorization device **56** may also use a conventional encryption key (f_{key}) to generate the second encrypted data stream from the permuted bits in accordance with the following expression:

$$R_{i+1} = f_{key} (H_i, H_{i-l}, ... H_{i-l-m})$$

where "l" and "m" are positive integers. Other conventional permuting operations and encryption keys may also be used and those described herein are provided as exemplary operations for generating an encrypted data stream.

To increase security and to inhibit the likelihood that the construction of the authorization device can be readily reverse-engineered, error insertion circuitry may be incorporated within the second stream encryptor **100** to intentionally insert "random" errors into the second encrypted data stream R. The presence of a limited number of intentional errors will typically increase the difficulty in reverse engineering the encryption operation by evaluating the data on the single-wire bus.

The first stream encryptor **82** within the deadman circuitry **54a** also preferably performs encryption operations to generate a third

encrypted data stream R' from the first data stream P and the second encrypted data stream R (with errors). In particular, the exemplary permuting and encryption key operations performed by the first stream encryptor **82** are preferably the same as the corresponding operations performed by the second stream encryptor **100** within the authorization device **56**. However, different operations may also be used by the stream encryptors in less preferred embodiments and the associated authorization detection circuitry may be considerably more complex.

The second and third encrypted data streams R and R' are evaluated at least periodically during the first time interval to determine whether a "match" is present between the authorization device **56** and the proprietary "software" loaded into the PLD **54**. This evaluation is preferably performed by the authorization detection circuitry ADC **84** within the PLD **54**. Thus, a direct ongoing comparison can be made between the encrypted data streams to determine whether there is a sufficiently close identity therebetween, while the PLD **54** is running the proprietary software.

Accordingly, as illustrated by FIG. 5, exemplary operations **110** for authorizing operation of a programmable logic device (PLD) may include operations to generate an at least weakly random data stream, Block **112**, and then generate first and second encrypted data streams within the PLD and authorization device, Blocks **114** and **116**. These encrypted data streams **118** are then evaluated, Block **118**. Exemplary evaluation operations may include a bit-by-bit comparison between the first and second encrypted data streams to determine if a sufficiently close match is present. Then, as illustrated by Block **120**, the PLD is disabled if the evaluation operation fails to indicate authorization of the PLD by the authorization device.

The above-described authorization scheme may also be applied to other forms of programmable logic devices (PLDs). Some of these programmable logics devices are frequently referred to by the acronyms PLAs, PALs, FPLAs, EPLDs, EEPLDs, LCAs, and FPGAs. In

addition, the preferred authorization scheme may be applied to application-specific integrated circuits (ASICs) that perform operations which are exclusively or at least partially hardware based. For example, the ASIC **58** of FIGS. 3B and 3C may be designed to perform a plurality of functions and operations useful for a variety of target applications. Customers purchasing such ASICs may be able to upgrade or expand the functions and operations performed by the ASIC **58** by purchasing one or more authorization devices at the time the ASIC **58** is purchased or thereafter. According to one embodiment applicable to ASICs, each of these additional authorization devices **56b**, **56c** can be connected to respective pins of the ASIC **58** to enable the ASIC **58** to perform additional or replacement functions and operations that correspond to the particular authorization device or combination of authorization devices. Alternatively, multiple authorization devices could be configured to share a common bus line, and in this case the respective ASIC **58** could be designed to have a set of chip selects for a plurality of authorization devices. The protocol could also be extended to deal with multiple authorization devices sharing a common bus line without individual selects.

Thus, an additional embodiment of the present invention may include first and second integrated circuit devices that generate first and second data streams, respectively, while the first integrated circuit device (e.g., ASIC, PLD) performs software and/or hardware controlled operations. This first integrated circuit device preferably has authorization detection circuitry therein that receives and at least periodically evaluates the first and second data streams and disables the software and/or hardware controlled operations when the first and second data streams fail to indicate a sufficient match between the second integrated circuit device (e.g., authorization device) and the software and/or hardware controlled operations performed by the first integrated circuit device. Still further embodiments of the present invention may include "slave" PLDs (or slave ASICs) that monitor the single-wire bus between a master PLD (or master

ASIC) and a respective authorization device. Each slave device may listen to the data provided on the single-wire bus to determine whether authorization is occurring. This determination may be made by incorporating into each slave device deadman circuitry that is similar to the circuitry within a corresponding master device. A slave device need not have circuitry to enable it to generate the first data stream P on the single-wire bus, however, additional circuitry may be necessary to enable it to operate in-sync with the communications between the master device and the authorization device.

In the drawings and specification, there have been disclosed typical preferred embodiments of the invention and, although specific terms are employed, they are used in a generic and descriptive sense only and not for purposes of limitation, the scope of the invention being set forth in the following claims.